Towards real time diffuse optical tomography:

Doulgerakis, Matthaios; Eggebrecht, Adam T; Wojtkiewicz, Stanislaw; Culver, Joseph; Dehghani, Hamid

DOI:
10.1117/1.JBO.22.12.125001

Towards real time diffuse optical tomography: Accelerating light propagation modeling employing parallel computing on GPU and CPU

Matthaios Doulgerakis[a], Adam Eggebrecht[b], Stanislaw Wojtkiewicz[a] Joseph Culver[b,c,d] and Hamid Dehghani[a]

[a] School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK

[b] Department of Radiology, Washington University School of Medicine, St. Louis, Missouri 63110, USA

[c] Department of Biomedical Engineering, Washington University in St. Louis, St. Louis, Missouri 63130, USA

[d] Division of Biology and Biomedical Sciences, Washington University School of Medicine, St. Louis, Missouri 63110, USA

Abstract—Parameter recovery in diffuse optical tomography is a computationally expensive algorithm, especially when used for large and complex volumes, as in the case of human brain functional imaging. The modeling of light propagation, also known as the forward problem, is the computational bottleneck of the recovery algorithm, whereby the lack of a real-time solution is impeding practical and clinical applications. The objective of this work is the acceleration of the forward model, within a Diffusion Approximation based finite element modeling framework, employing parallelization to expedite the calculation of light propagation in realistic adult head models. The proposed methodology is applicable for modelling both continuous wave and frequency domain systems with the results demonstrating a tenfold speed increase when GPU architectures are available, whilst maintaining high accuracy. It is shown that for a very high resolution finite element model of the adult human head with ~600,000 nodes, consisting of heterogeneous layers, light propagation can be calculated at ~0.25 seconds per excitation source.

Keywords—NIRFAST, Parallel Computing, GPU, Diffuse Optical Tomography, Finite Element Method

1. INTRODUCTION

Functional neuroimaging provides an essential tool in the study of the brain. It has been used to detect, localize, and classify brain activations during physical and psychological events, propelling applications in a myriad of areas, such as guiding treatment and monitoring the rehabilitation progress in cases of stroke, depression, or schizophrenia [1–3]. Diffuse optical

tomography (DOT) is a soft tissue imaging technique based on injecting near-infrared (NIR) light in a volume and measuring the re-emerging light. In brain related studies, the measured alterations in the attenuation of the resurfacing NIR light reflect changes in blood oxygenation and concentration induced by tissue metabolism within the brain due to local neuron activations. Therefore, DOT has been used for functional brain imaging[4,5], neonatal brain monitoring [6,7], as well as for measuring absolute oxygenation values in the brain[8]. NIR light is non-ionizing, and requires relatively low-cost equipment, which is wearable and therefore allows some movement of the subject. Additionally, DOT is relatively portable and can be used in clinical applications where use of fMRI or PET is not possible, for example as a bedside monitoring tool.

DOT specifically is concerned with tomographic reconstruction of volumetric and spatially distributed optical parameters from finite boundary measurements. This is commonly solved as an optimization problem using model-based approaches, whereby accurate modeling of light propagation within the volume, known as the forward model, is required. Therefore, to allow a real-time parameter recovery from measured data, both fast and accurate forward modeling is essential.

The objective of this work is the acceleration of the forward light propagation model, while maintaining numerical accuracy. Specifically, the focus is on the application of DOT for functional imaging on adult human head, employing the finite element method (FEM) to solve the diffusion approximation (DA) for modeling of light propagation as implemented within the NIRFAST[9] modeling and image reconstruction software package.

The proposed acceleration approach relies on employing parallel computing to expedite the solution of the forward problem, an option that has recently become popular due to the relatively low cost of GPUs and that has attracted the attention of researchers for solving similar problems in medical imaging[10–12]. In DOT the acceleration of the forward model with GPUs has been employed for Monte-Carlo algorithms[13–15], where simulating the behavior of each photon can

be efficiently parallelized, with reported accelerations in the scale of $10^2 - 10^3$. However, millions of photons must be simulated to achieve an accurate solution, so modeling the total fluence for a geometrically large volume and multiple excitation sources still requires time in the order of tens of minutes.

Accelerating the forward model of DOT using GPU parallelization has also been reported with FEM formulation. Specifically, the acceleration of the forward model solution has been proposed for frequency domain simulations[16]. Unfortunately, due to a lack of sparse arithmetic architecture, computationally tractable mesh sizes have been limited to ~9,000 nodes, due to the excessive memory requirements. Solving the forward model employing GPU parallelization, in continuous wave, with parallelization over the nodes and over the excitation sources simultaneously, using a block based formulation of the forward linear system has also been proposed[17]. However, due to the size of the augmented block matrix, this approach is only applicable in small size meshes, with up to ~2,500 nodes. Using multiple GPUs to solve the forward problem in continuous wave, for infant brain studies has been suggested[18] but has been only evaluated qualitatively in a homogenous phantom head model, again in meshes with ~9,000 nodes. An approach combining CPU and GPU parallelization was proposed[19] where the imaging domain is decomposed into overlapping subdomains, therefore allowing a high level of parallelization for the forward problem. However, this approach was only evaluated in continuous wave, on a simplified cylindrical geometry with homogenous optical properties where the decomposition in regular overlapping subdomains is a straightforward procedure; while in the case of a complex volume, as the adult head, such decompositions are not a trivial task. Finally, a framework for the solution of the forward problem in continuous wave and frequency domain, accelerated in the GPU, was proposed and evaluated in homogenous cylindrical models with up to 330,000 nodes[20]. It was shown that relation between error of the iterative solution and optical properties of the volume was identified while the single precision

numerical accuracy was found to be insufficient for solving for a wide range of optical properties. However, none of the previous work has evaluated the accuracy and computational speed of iterative solvers on anatomically realistic head models, with high resolution meshes and high density (HD) DOT system.

This work provides tractable solutions to overcome the current computational time and memory limitations arising when dealing with high resolution FEM meshes and HD source-detector (SD) pairs, both important features for high quality functional DOT (fDOT) brain imaging. Additionally, an extended evaluation is performed, on high resolution meshes with up to ~600,000 nodes, based on a realistic anatomical head model, with five tissue layers, focusing in achieving the desired numerical accuracy for functional brain imaging. Furthermore, support for complex numbers for the cases of frequency domain simulations is incorporated. Specifically, section 2 outlines a DOT implementation using the NIRFAST package along with details highlighting the computational complexity of parameter recovery, and the proposed parallelization approaches. In section 3 the results are presented and discussed in the context of employing iterative solvers for the forward problem in DOT. Section 4 concludes with the remaining challenges and opportunities for further optimizations and applications.

2. METHODOLOGY

The procedure followed in DOT image reconstruction can be summarized by the following consecutive steps: modelling the light propagation through the medium, also known as the forward problem and a parameter recovery process based on the forward model and NIR measurements, also known as the inverse problem. This section provides an overview of the underlying mathematics that directly affect the computational aspects of DOT and emphasizes the necessity of parallelization, specifically considering the existing implementation within NIRFAST. Currently, the most computationally expensive procedure is the solution of large FEM sparse linear systems, involved in estimating light propagation in the forward problem.

2.1. Forward problem

The first step of the DOT algorithm, the forward problem, is the basis for the application of model-based image reconstruction therefore it must be as accurate as possible, numerically and geometrically, as any errors will affect the formulation of the inverse problem.

The accuracy of the numerical solutions of FEM is greatly affected by the prior knowledge of the underlying tissue geometry; therefore the maximum potential is reached when FEM is combined with input from other standard imaging techniques [21,22] or generic atlas

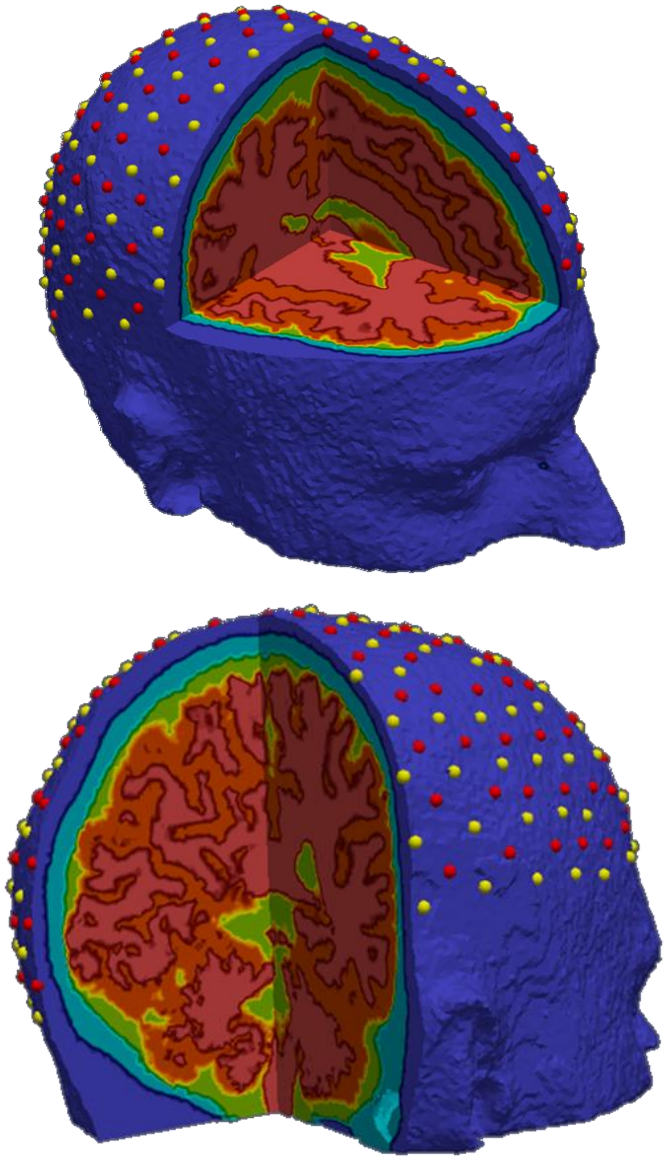

Fig. 1 The modeled high-density DOT system with 158 sources (red) and 166 detectors (yellow), on an adult head model with 5 tissue layers.

models [23]. Mesh generation based on structural images from other modalities, usually MRI, is a well-studied problem. There are existing algorithms that automatically segment tissue layers and create surface-based meshes [24]. When the model volume remains constant, but meshes with more elements are created, what effectively changes is the resolution of the mesh. Higher resolution meshes have elements with smaller volume and therefore minimize partial volume effects due to mesh elements integrating over multiple segmented tissue regions. The fine complex structures, such as the brain cortex in a head model, can be modeled more accurately with a high-resolution mesh, providing optical properties assigned to each node (or element) that are more likely to represent the underlying baseline optical properties of the tissue at each position.

When the volume is meshed, FEM is employed to formulate a discretized weak form representation of the DA for each node of the mesh. It follows that as the volume of each element tends towards zero (increasing the mesh resolution), the calculated approximation becomes more accurate, therefore in fDOT very high resolution meshes, with up to 600,000 nodes are used (Table 1). This is a domain size dependent problem, a smaller volume such as an infant's head, will require fewer nodes to achieve elements of sufficiently small volume. Additionally, dividing the volume into a higher resolution mesh minimizes the discretization error introduced by FEM. However, increasing the mesh resolution results in the requirement of solving a bigger linear system to estimate the light fluence, which, until now, has dramatically increased computational time. The focus of this work is optimizing numerical approaches employing FEM-DA to estimate light propagation - a well-studied problem that assumes the DA is valid for all tissue properties used [9,25,26]. However, the advancements described herein can be applied to any models (e.g. Radiative Transport Equation) based on discretized approximations.

FEM is a numerical technique where a heterogeneous problem is divided into many smaller parts, creating a non-uniform mesh, consisting of elements defined by connected nodes. The diffusion equation can then be discretized and represented as a set of linear equations, describing simultaneously all the nodes and hence the entire medium. The problem thus reduces to a sparse, well-posed linear problem of the form:

$$\boldsymbol{M\Phi = q} \tag{1}$$

where **M** is a sparse matrix with dimensions N by N, with N denoting the number of nodes; **q** represents the sources and has dimension N by the number of sources Q of the DOT system, and **Φ** is the photon fluence rate for all nodes for each source, as has dimensions N by Q.

Table 1 Different resolution meshes based on linear tetrahedral elements for an adult head model

Number of nodes	Number of elements	Element volume average and standard deviation (mm^3)
50721	287547	9.26 ± 3.43
68481	393863	6.76 ± 2.29
101046	589658	4.51 ± 1.64
139845	821926	3.24 ± 1.18
205568	1215434	2.19 ± 0.78
235564	1395242	1.90 ± 0.68
271135	1609152	1.65 ± 0.59
305058	1813036	1.46 ± 0.52
324756	1931374	1.37 ± 0.49
360777	2149250	1.23 ± 0.43
411567	2454350	1.08 ± 0.37
515837	3084689	0.86 ± 0.29
610461	3656890	0.72 ± 0.24

2.2. Inverse problem

The second step of DOT image reconstruction, the inverse problem, estimates tissue optical properties based on the forward model and NIR boundary measurements. To acquire the required measurements, NIR light is injected into the imaging volume by optical fibers or LEDs

positioned on its surface (light source) while the transmitted and reflected diffused light is measured, using optical fibers or detector arrays (light detectors), as it remerges from the surface of the volume in nearby positions.

There are three categories of NIR measurement systems: continuous wave (CW), frequency domain (FD) and time resolved (TR) [27]. TR and FD systems are advantageous because the measured transient time or phase shift information allows the determination of scattering and absorption simultaneously, while CW systems are unable to make quantitative absorption measurements without a-priori assumptions of the scattering [28]. However, modeling of light propagation in FD has increased computational cost due to complex arithmetic, whilst TR requires multiple light propagation models to be estimated for consecutive time instances.

DOT acquires boundary data from multiple and overlapping SD pairs, therefore provides valuable spatial depth information, and improves lateral image reconstruction resolution. Studies have shown that the density of the SD pairs can directly affect the spatial resolution and localization accuracy of reconstructed images [29,30]. HD-DOT, an arrangement with dense SD pairs, is considered to produce superior results and is particularly effective in brain functional imaging [4,7,29,31–33].

In fDOT, models of estimated light propagation based on assumptions of the underlying tissue scattering and attenuation are used to create a sensitivity matrix, also known as the Jacobian. The Jacobian is the basis for solving the inverse problem, allowing the recovery of spatiotemporal changes of internal optical properties for the whole volume, using temporal derivatives of measurements obtained on the surface of the volume, known as boundary data, then performing single step (linear) reconstruction.

The approach employed to form the Jacobian is the adjoint method[34], where the direct fluence for each source and the adjoint fluence for each detector must be calculated; then the

sensitivity is calculated as the product of the direct and the adjoint field, with regard to the basis function for each element, the result of which is a dense matrix. The construction of the Jacobian therefore requires the forward problem to be solved twice; once for all sources and once for all detectors.

2.3. Computational problem

The solution of large sparse linear systems involved in the forward model is currently the computational bottleneck of the DOT algorithm. In this presented example, the fluence throughout the volume, must be calculated for all 158 sources and for all 166 detectors to create the Jacobian for the modeled DOT system illustrated in Fig.1. To solve the linear systems arising in the forward modeling, in the form $\mathbf{Ax} = \mathbf{b}$, for $\mathbf{x}$, where $\mathbf{A}$ has dimensions $N \times N$, where N is the number of uknowns, the inverse of $\mathbf{A}$ must be calculated. However, calculating a matrix inverse is computationally inefficient, therefore a variety of algorithms have been proposed that can solve linear systems without explicitly calculating a matrix's inverse. These algorithms can be either direct, providing an exact solution, or an approximate, usually employing an iterative algorithm. The storage convention used in this work to represent sparse matrices within memory is the compressed row storage, which requires $2N_{NZ} + N + 1$ space in memory for a $N \times N$ matrix with N_{NZ} non-zero entries.

2.3.1.Direct solvers

The most popular direct solver is the Gaussian Elimination, also known as row reduction, where the echelon form of $\mathbf{A}$ is calculated through row operations on the augmented matrix $(\mathbf{A}|\mathbf{b})$. The echelon form of $\mathbf{A}$ is an upper triangular matrix, making the solution of the linear system easy through backward substitution. However, Lower Upper (LU) factorization is considered the standard efficient computational approach for direct solution of a linear system. The LU factorization decomposes $\mathbf{A}$ into lower ($\mathbf{L}$) and upper ($\mathbf{U}$) triangular matrices. Substituting $\mathbf{L}$ and $\mathbf{U}$ in $\mathbf{Ax} = \mathbf{b}$ gives $\mathbf{LUx} = \mathbf{b}$, and letting $\mathbf{Ux} = \mathbf{Y}$, then $\mathbf{LY} = \mathbf{b}$. Now it is trivial to solve $\mathbf{LY} =$

b for **Y** through forward substitutions and then solving $\mathbf{Ux} = \mathbf{Y}$ through backwards substitutions. The advantage of LU factorization over the traditional Gaussian elimination is that decomposing **A** into **L** and **U** is independent of **b**, also known as the right-hand-side vector. This allows **L** and **U** to be used for solving for multiple right-hand side vectors. However, this approach has very large memory requirements of $N^2 + N$ and high computational cost, of $\frac{2}{3}N^3$ floating point operations (FLOPS), to solve a full linear system. A more efficient alternative, that can be used only if **A** is Hermitian, therefore symmetric when real, is the Cholesky factorisation, where **A** is decomposed to $\mathbf{L}\,\mathbf{L}^*$, where * denotes the transpose conjugate operator, requiring $\frac{1}{2}N^2 + N$ memory for a full system. The linear system can be solved as with the LU method, substituting $\mathbf{U} = \mathbf{L}^*$, with computational cost, for the solution of full systems, of $\frac{N^3}{3}$ FLOPS. However, in the case of sparse linear systems, such as resulting from the FEM formulation, memory and computational costs are related to the number of non-zero elements of **A** rather than the size N and, additionally, there are reordering strategies that when applied on sparse matrices allow more sparse factorizations. Specifically, in this work, the approximate minimum degree permutation algorithm was found to produce the most sparse factorizations, therefore was used for all the direct solvers. Nevertheless, factorization approaches rely on forward and/or backward substitutions to provide a solution, therefore they cannot be efficiently parallelized. In MATLAB when solving linear systems invoking the backslash operator, the Cholesky approach is used when the matrix is Hermitian, otherwise the LU approach is employed. The "spparams"[35] command was used to confirm that all the real linear systems were solved with Cholesky solver and all the complex with LU. The MATLAB backslash operator is considered as the numerical ground truth for the solution of linear systems throughout this work.

2.3.2.Iterative solvers

To overcome the computational limitations of direct solvers a variety of approximate solvers have been proposed that can be classified into three general categories: iterative, multigrid, or domain decomposition methods[36]. Multigrid and domain decomposition methods can be very efficiently parallelized, with solving speed not greatly affected by the size of the linear system. However, these methods require additional input parameters (e.g., the range of eigenvalues of the system, restriction and prolongation parameters, smoothing operators) that might be difficult to define and may vary for different systems to efficiently converge to adequate approximations. As such, these methods work best when they are tailored to solve a very specific problem. In contrast, iterative solvers are generic, and require little or no additional input from the user, and thus are traditionally chosen for the solution of linear systems describing light propagation.

Iterative approaches approximate a solution vector $\mathbf{x}_n$ and then attempt to minimize the residual $r_n = \|\mathbf{b} - \mathbf{A}\mathbf{x}_n\|$ through n iterations, until r_n is lower that a user defined residual threshold r_{th}. However, in practice, the termination criteria are defined relatively as $t_{\text{c}} = \frac{r_n}{r_0}$, where r_0 is the residual after the initial guess, with the initial guess $\mathbf{x}_0$ set usually as a vector of zeros. Using a relative threshold ensures that the iterations will converge with a final r_n usually within the same order of magnitude as the t_{c}, even when the number of unknowns is very large. Iterative approaches usually work on a projection space for increased computational efficiency. The most established projection scheme is the Krylov subspace, which is based on the Cayley-Hamilton theorem that implies that the inverse of a matrix can be found as a linear combination of its powers. The Krylov subspace generated by a $N \times N$ matrix $\mathbf{A}$, and a vector $\mathbf{b}$ of dimension N, is the linear subspace spanned by images of $\mathbf{b}$ under the first α powers of $\mathbf{A}$.

$$K_\alpha(\boldsymbol{A}, \boldsymbol{b}) = span\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \boldsymbol{A}^2\boldsymbol{b}, \dots, \boldsymbol{A}^{\alpha-1}\boldsymbol{b}\} \quad (2)$$

This formulation avoids matrix to matrix operations, and instead utilizes matrix to vector operations which can be very efficiently implemented in parallel architectures. The Krylov

subspace is generated while the solver seeks to find the minimum of the projection space. Usually least square or gradient based optimization techniques are employed to solve such problems. There are many proposed algorithms to implement a Krylov space solver but there is no clear conclusion on which one is fastest when the same termination criteria is required[37]. The most popular approaches for the Krylov space gradient optimization is the Conjugate Gradient (CG), but is not guaranteed to work in non-Hermitian linear systems[38]. However, there are appropriate Krylov subspace solvers that can handle non-Hermitian systems with relatively low additional computational cost, like the Biconjugate Gradient Stabilized (BiCGStab).

2.3.3.Preconditioners

Iterative solvers do not have robust performance, and can be very slow, when the condition number of the system is very large. To overcome this, preconditioned versions of the solvers have been developed. Efficient preconditioning can largely reduce the condition number of a linear system leading to a dramatically reduced number of iterations to convergence. The preconditioner **P**, in effect is changing the geometry of the Krylov subspace to a simpler one, making the solution of the system much easier by providing an approximation of the matrix inverse that is easy to compute and solve. Instead of trying to minimize $\|\mathbf{b} - \mathbf{A}\mathbf{x}_n\|$,the expression to minimize becomes $\|\mathbf{P}^{-1}\mathbf{A}\mathbf{x}_n - \mathbf{P}^{-1}\mathbf{b}\|$, to be effective, the preconditioner **P** must be of much lower condition than **A**. In general, the $\mathbf{P}^{-1}\mathbf{A}$ product should be as close as possible to identity matrix or in other words $\mathbf{P}^{-1} \approx \mathbf{A}^{-1}$. It is hard to theorize what consists a good preconditioner, the main diagonal of **A**, also known as Jacobi preconditioner, can be very effective in diagonally dominant systems, however, usually an incomplete factorization of **A** is used; as incomplete LU or incomplete Cholesky (IC) factorization. Recent research on solving linear systems focuses mainly on the choice of efficient preconditioners, emphasizing preconditioners that can be implemented in parallel architectures [39], rather than improving the solvers themselves. This happens because the time within each iteration is greatly reduced due

to the high level of parallelism offered by GPUs[40], whilst preconditioning reduces the number of iterations needed to converge. Nevertheless, in practice, choosing the best preconditioner is usually a trial and error procedure. There are block-based preconditioners that are favourable for GPU parallelization[41,42]. In practice, the IC with no prior permutation or pivoting scheme was found to be the best preconditioning option for fast convergence of the MATLAB based iterative solvers, whilst the Factorized Sparse Approximate Inverse (FSAI)[43] was found to be the option that produced the fastest overall result with the CUDA and OpenMP implementations. FSAI is constructed by solving local linear systems for each column of $\mathbf{A}$ to approximate an $\mathbf{A}^{-1}$with sparsity pattern defined by powers of $\mathbf{A}$. Additionally, a preconditioner inspired by FSAI was implemented, where the local linear systems were solved in parallel and only for the three larger values for each column, achieving similar preconditioning effectiveness whilst reducing the computational time for the construction of the preconditioner. This preconditioner is referred to as "FSAIP" for the rest of this work.

2.3.4.Numerical accuracy

The iteration residual r_n, and to an extent, the realization of the termination criteria t_c, is bound to the numerical binary representation precision of numbers that the machine, the programing language and the employed libraries allow. In modern systems, this is double precision, represented by 64 bits of memory, which in practice can represent numbers with relative differences no smaller than 2^{-52}, this is ~2.22×10^{-16}, which is the minimum value defined in MATLAB. Any difference smaller than this is lost due to the quantization involved in converting a number that belongs to the real set $\mathbb{R}$ into the binary set $\mathbb{B}_2^{64}$, where $\mathbb{B}_2 = \{0,1\}$. Therefore, requesting termination residual lower than a scale of 10^{-16}, will not result in a more accurate solution, since any additional variation would be under the double precision quantization bin size of MATLAB and will be rounded to the nearest bin. Apart from the binary rounding errors, when solving a linear system with an iterative solver, the maximum solving

precision that can be achieved is analogic to the condition number of the system. The condition number of a linear system $\mathbf{A}$ can be estimated as $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, where $\|\ \ \|$ is a matrix norm (see section 2.6). The condition number of a linear system is large if there are big differences in its eigenvalues, therefore when solving large condition linear systems with iterative solvers good preconditioning is essential to achieve convergence to low errors.

2.3.5. Complex numbers support

The existing open source libraries that provide low-level functions and primitive data structures for parallel programming support on CPU and GPU are the Open Multi-Processing language (OpenMP)[44] and Compute Unified Device Architecture (CUDA)[45]. However, up to the current version, OpenMP 4.5, does not provide native complex numbers support; and CUDA, whilst it provides support for complex numbers, does not come with high level mathematical functions such as sparse iterative linear solvers and preconditioners, therefore implementations that allow complex support are not a trivial task. Additionally, open source mathematical libraries that provide iterative solving of sparse linear systems on parallel architectures, as PARALUTION[46] and ViennaCL[47], do not provide complex numbers support. However, when formulating the forward problem for systems operating in the frequency domain, the resulting linear system consists of complex numbers. Nevertheless, there are algebraic schemes that allow a linear system of complex numbers to be represented as an equivalent system of real numbers, solved in the real number domain, and then the solution can be converted back to a complex representation. There are four possible formulations of equivalent real systems as described in [48], the approach chosen for this work is the K1 approach, that is formulates as:

$$\mathbf{A}_\mathrm{c} = (x + yi) \leftrightarrow \begin{pmatrix} x & -y \\ y & x \end{pmatrix} = \mathbf{A}_\mathrm{r} \tag{3}$$

where $\mathbf{A}_\mathrm{c}$ is the complex form and $\mathbf{A}_\mathrm{r}$ the equivalent real representation generalising, the n^{th} dimensional complex linear system $\mathbf{A}_\mathrm{c}\mathbf{x}_\mathrm{c} = \mathbf{b}_\mathrm{c}$ with entries:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{4}$$

is equivalent to the real linear system $\mathbf{A}_\mathrm{r}\mathbf{x}_\mathrm{r} = \mathbf{b}_\mathrm{r}$ with entries:

$$\begin{pmatrix} \Re a_{1,1} & -\Im a_{1,1} & \Re a_{1,2} & -\Im a_{1,2} & \cdots & \Re a_{1,n} & -\Im a_{1,n} \\ \Im a_{1,1} & \Re a_{1,1} & \Im a_{1,2} & \Re a_{1,2} & \cdots & \Im a_{1,n} & \Re a_{1,n} \\ \Re a_{2,1} & -\Im a_{2,1} & \Re a_{2,2} & -\Im a_{2,2} & \dots & \Re a_{2,n} & -\Im a_{2,n} \\ \Im a_{2,1} & \Re a_{2,1} & \Im a_{2,2} & \Re a_{2,2} & \cdots & \Im a_{2,n} & \Re a_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \Re a_{n,1} & -\Im a_{n,1} & \Re a_{n,2} & -\Im a_{n,2} & \cdots & \Re a_{n,n} & -\Im a_{n,n} \\ \Im a_{n,1} & \Re a_{n,1} & \Im a_{n,2} & \Re a_{n,2} & \cdots & \Im a_{n,n} & \Re a_{n,n} \end{pmatrix} \times \begin{pmatrix} \Re x_1 \\ \Im x_1 \\ \Re x_2 \\ \Im x_2 \\ \vdots \\ \Re x_n \\ \Im x_n \end{pmatrix} = \begin{pmatrix} \Re b_1 \\ \Im b_1 \\ \Re b_2 \\ \Im b_2 \\ \vdots \\ \Re b_n \\ \Im b_n \end{pmatrix} \tag{5}$$

The equivalent real system has the same sparsity pattern and sparsity factor as the original complex system, however the new system has double the time of unknowns, therefore requires double the computations, and additionally FEM-DA linear systems in FD are no longer Hermitian, therefore the BiCGStab solver is employed for FD simulations.

In addition, a BiCGStab solver, based in CUDA, operating directly on the complex domain was implemented and used with the developed parallel constrained FSAI version (FSAIP) to solve FD simulations.

2.4. GPU/CPU parallelization

The proposed approach for accelerating fluence estimation relies on employing efficient libraries for linear algebra operations, and performs remarkably faster when GPU based parallel architectures are available. Over the last decade, the technical advancements in GPUs, and their relatively low cost, has made GPU computing a very attractive option. Specifically, many linear algebra operations can be parallelized very efficiently in GPU architectures[49] while using sparse representations, resulting in massive reductions of computational time. This can be applied on the solution of the forward model, dramatically decreasing the computational time required to estimate the Krylov subspace. Solvers based on libraries that can be used both in CPU and GPU

were implemented, to guarantee accessibility by all users. Additionally, the solvers were compiled as MATLAB executable files (mex files) for both Windows and Linux 64 bit systems, to allow easy invocation from within the MATLAB environment where the NIRFAST package is based. Specifically, to take advantage of the GPU computing power, implementations based on CUDA[45] were produced, which require the CUDA runtime that is provided with the NVIDIA drivers, and the CUDA Software Developer Kit (SDK) that is free to download. For CPU environments the OpenMP backend was used, that is also publicly available with all standard C/C++ compilers. The mathematical library employed to provide efficient implementations of high level linear algebra operations is PARALUTION[46] which offers a wide variety of linear solvers and preconditioners, supports sparse matrix and vector formats, and allows a high level of abstraction between code and hardware, making the code highly portable and efficiently scalable to the available hardware. The produced CUDA based implementations will retreat to OpenMP, if there is no GPU available in the system.

An algorithm to distribute workload between the CPU and GPU was implemented, the workload was distributed by balancing the right-hand side input (sources) between CPU and GPU. Benchmarking tests were performed on all mesh resolutions to define the best workload distribution in each case. However, it was found that in all meshes above 70,000 nodes the solely GPU based solution was faster, whilst with meshes with smaller number of nodes (~50,000), the computational time reduction was less than a second. On the other end, the CPU implementation is faster than the equivalent GPU implementation in meshes with less than 15.000. This is primarily due to time consuming data transfer and device initialisation procedures. Nevertheless, this is depended on hardware, number of right-hand side vectors and complexity of the imaging domain.

MATLAB provides sparse linear solvers on the CPU, that can be easily parallelized over the right-hand side vectors using the parallel computing toolbox. Though, there are overhead data

transfers between memories (RAM, CPU cache memory, GPU memory) and between computational threads and memory that do not allow computational accelerations to scale linearly with the number of available computational cores.

2.5. Experimental set-up

Data from MRI of an adult head was segmented and meshed into thirteen different resolution meshes using the algorithm proposed by Jermyn et al[24]. The modeled DOT instrument is a high-density system with 158 NIR light sources and 166 detectors. Each detector is related to sources in separation distance configurations from 1.3 to 4.6cm, resulting to 3500 associated source-detector pairs. More details about the resolution of the meshes can be found in Table 1 and the optical properties for each layer of the anatomical model are described in Table 2 [32].

Table 2 Optical properties of tissue layers at 750nm wavelength [32]

Tissue Layer	μ_a (mm^{-1})	μ_s' (mm^{-1})	Refractive Index
Scalp	0.0170	0.74	1.33
Skull	0.0116	0.94	1.33
Cerebrospinal fluid	0.004	0.3	1.33
Gray Matter	0.0180	0.84	1.33
White Matter	0.0167	1.19	1.33

The light propagation model was calculated for all 158 sources in all experiments, in continuous wave mode and in frequency domain mode at a modulation frequency of 100 MHz, for one NIR wavelength of 750nm. All the experiments were performed on a desktop computer with 16GB of RAM, an Intel Core I7-4790 CPU with 4 physical cores, allowing two threads per core, resulting to 8 logical cores @ 3.6GHz, and a NVIDIA GTX970 graphics card with 1664 logical cores @ 1050MHz with 4GB dedicated memory.

2.6. Metrics

It is important to ensure that employing an iterative linear solver will not increase the error of the solution. To this end, the accuracy of the proposed solvers was compared against the direct solution, calculated with the backslash operator in MATLAB. There is no standard way of comparing two matrices, $\mathbf{\Phi}_{\mathrm{ref}}$, for the fluence calculated with a direct solver, and $\mathbf{\Phi}_{\mathrm{ite}}$, for the fluence calculated with an iterative solver, however the first step for all approaches is taking the difference $\mathbf{\Phi}_{\mathrm{dif}} = |\mathbf{\Phi}_{\mathrm{ite}} - \mathbf{\Phi}_{\mathrm{ref}}|$. The most common metrics to quantify the difference $\mathbf{\Phi}_{\mathrm{dif}}$ are the ones induced from vector norms: the 1-norm $\|\mathbf{\Phi}_{\mathrm{dif}}\|_1$ which is the maximum of the column sums of $\mathbf{\Phi}_{\mathrm{dif}}$, the ∞-norm $\|\mathbf{\Phi}_{\mathrm{dif}}\|_\infty$ which is the maximum of the row sums of $\mathbf{\Phi}_{\mathrm{dif}}$, and the l2-norm $\|\mathbf{\Phi}_{\mathrm{dif}}\|_2$ which is the maximum singular value of $\mathbf{\Phi}_{\mathrm{dif}}$, also known as the spectral norm.

However, those metrics do not provide easily comprehensible quantities, therefore the relative error per node r,was calculated as:

$$\varepsilon_{rel}(r) = \frac{\left|\boldsymbol{\Phi}(r)_{ref} - \boldsymbol{\Phi}(r)_{ite}\right|}{\left|\boldsymbol{\Phi}(r)_{ref}\right|} \times 100\% \quad (6)$$

This relative error representation is useful for visualisation of the error on the mesh nodes and boundary data, and provides more comprehensible numbers than the matrix norms.

3. RESULTS AND DISCUSSION

The evaluation is performed in one adult head model using a HD-DOT system with 158 sources and 166 detectors (Fig. 1). The behaviour of the solvers is examined under varying error demands, and in different mesh resolutions, considering the accuracy and the computational time. The focus is on the relation between mesh resolution (and hence problem size), termination criteria, computational time, and solution error. The direct solutions are only possible to calculate up to the 400,000 nodes mesh for continuous wave (CW) and up to 200,000 nodes in frequency domain (FD) systems, due to high memory requirements, so all quantitative comparisons are performed in the subset of the meshes where a direct solution is available.

3.1. Qualitative and quantitative comparisons

Considering that there is already some error introduced by the discretization of the diffusion approximation within the FEM formulation, the error from solving the linear systems should be kept at a minimum. However, the amount of error that can be afforded in the modelling procedure is dependent to the error tolerance for the application. **Error! Reference source not found.** shows the surface fluence when utilizing the CUDA based solver at different termination criteria; the simulated light source is near the back of the head, indicated by the blue dot and arrow.

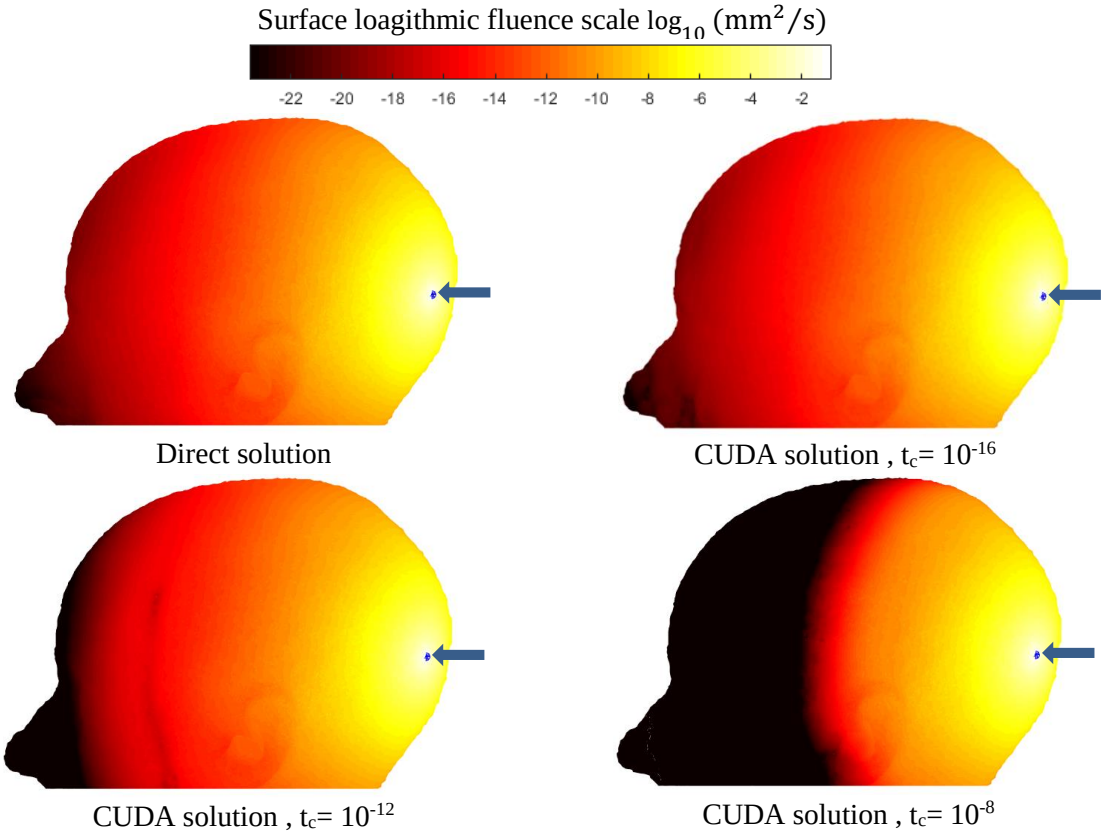

Fig. 2 Visual comparison of surface fluence whilst using different termination criteria. Simulation in continuous wave, for one source indicated in blue, in a 400,000 nodes mesh, solving with CUDA CG.

When high termination criteria are set, the fluence is not estimated for the distant nodes as the solution iterates to a stable solution quickly. The fluence approximately follows an exponential decay through tissue, therefore its value dramatically decreases with distance from

the source, therefore the required termination criteria are reached while only partially calculating the solution for the highest fluence values.

As FD simulations provide amplitude and phase information, the errors for each were examined separately. Fig. 3 and 4 provide a quantification of the relationship between distance from the source and the relative nodal amplitude and phase errors arising from solving with an iterative solver. The demonstrated simulation is in the frequency domain, at 100MHz, for a mesh with 200,000 nodes, solving with a CUDA BiCGStab & FSAI. In Figs 3 and 4, the maximum relative error for all nodes as a function of distance from the source is extracted for different termination criteria.

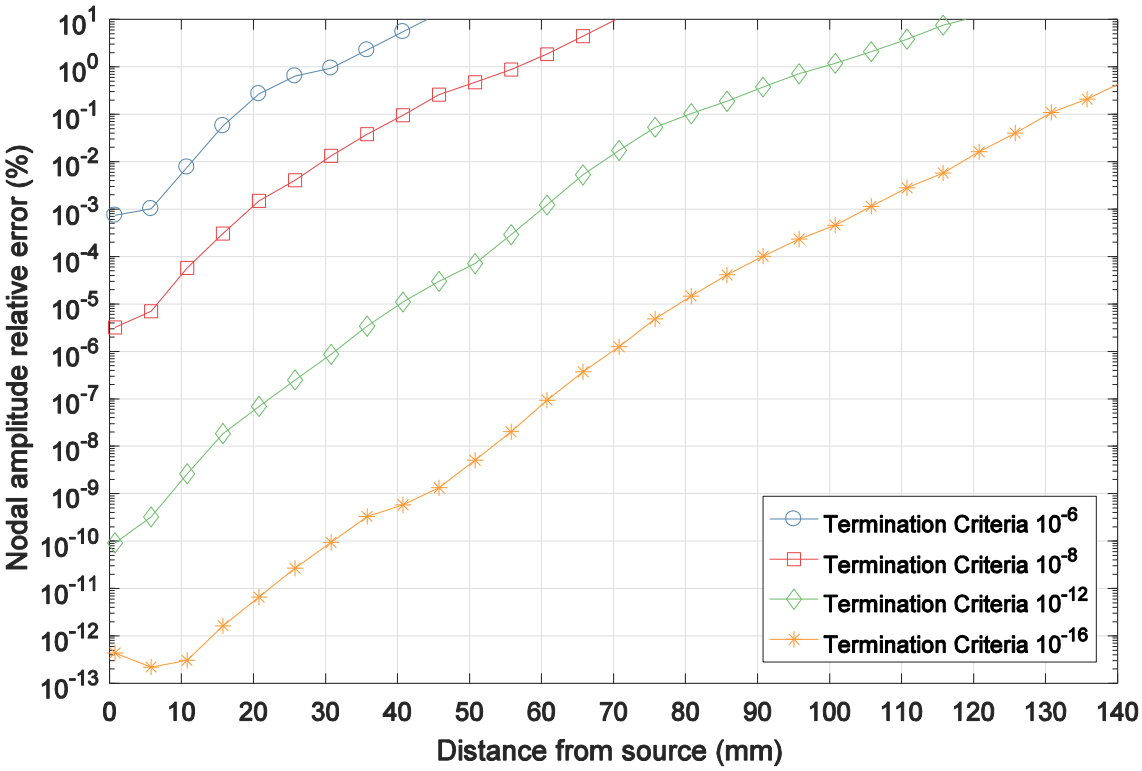

Fig. 3 Maximum relative amplitude errors per node (Eq.6) as a function of distance from source. Comparison between different termination criteria. Simulation 100MHz frequency, on a 200,000 nodes mesh, solving with CUDA BiCGStab.

As evident, the relative errors are small and located away from the source with low termination criteria of 10^{-16}, but they become larger and manifest nearer the source as the termination criteria rises. Similar results (not shown) were acquired for amplitude errors from continuous wave simulations.

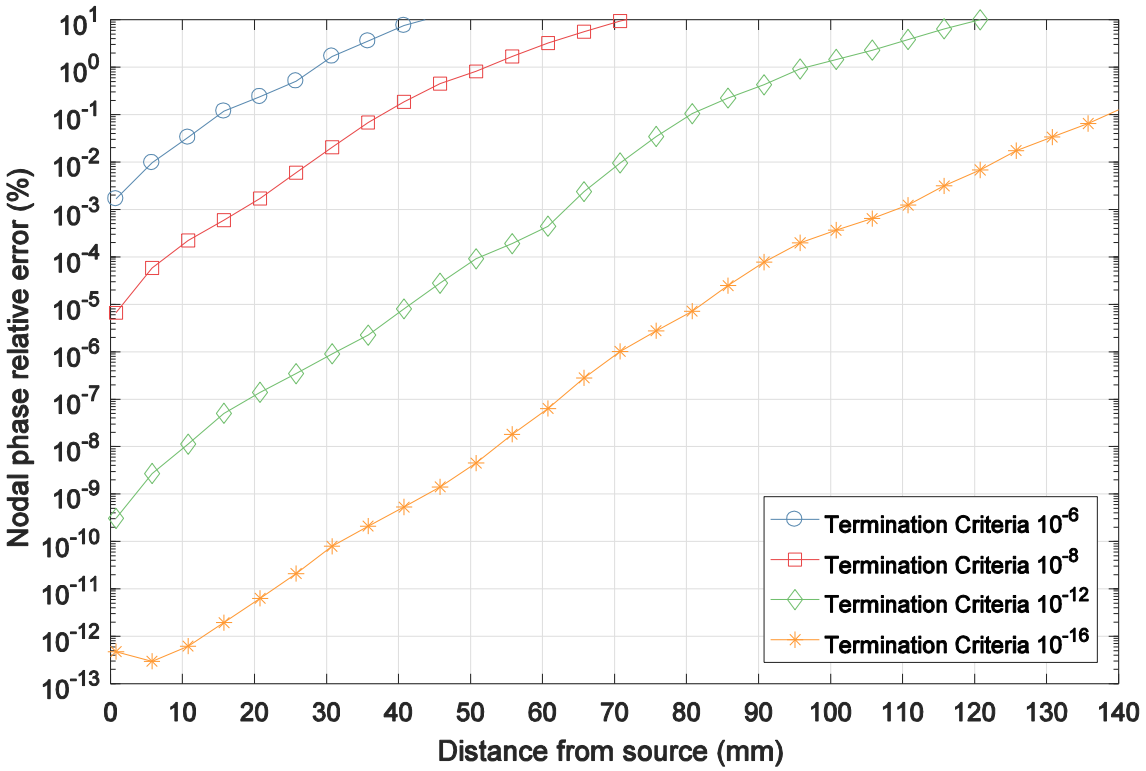

Fig. 4 Maximum relative phase errors per node (Eq.6) as a function of distance from sources. Comparison between different termination criteria. Simulation 100MHz frequency, on a 200,000 nodes mesh, solving with CUDA BiCGStab.

Lower termination criteria will provide smaller numerical errors, but also slower solver convergence, as a larger number of iterations is required. In the modelled DOT system, the maximum SD separation typically considered to acquire boundary data is at 46 mm. The performed evaluation in Figs 3 and 4 reveal that for anatomically accurate adult head models, the termination criteria can be selected in the range of 10^{-8} or lower, for CW and FD systems, to ensure that minimal error is introduced in the parameter recovery, when acquiring measurements from SD separation distances less than 46 mm. The sensitivity matrix will have approximately square of the error of the forward solution. Therefore, termination criteria chosen to be large, a practise often employed to accelerate reconstructions, while the boundary data is measured in large SD separations, can lead to large errors in the sensitivity matrix and, consequently, large errors in the parameter recovery and image reconstruction.

3.2. Computational Time Comparisons

Three parameters mainly affect the computational speed of iterative linear solvers: the size of the problem, which in our case is the number of nodes, the number of right hand side vectors, which in our case is the number of excitation sources, and finally the termination criteria. All the experiments were performed 10 times, the mean time is shown in all figures, while the standard deviation in all cases was small, at around 1 second for CPU implementations and 0.1 second for GPU, therefore is not shown in the figures.

Fig. 5 shows the computational time for fluence estimation for 158 sources in a 400,000 noded mesh as a function of termination threshold. The direct solver provides an exact solution to the linear system, therefore does not introduce any error. However, is displayed as a horizontal line through all the termination criteria in Fig. 5, to serve as point of reference. The GPU based solver yields the best termination criteria to computational time ratio. Employing

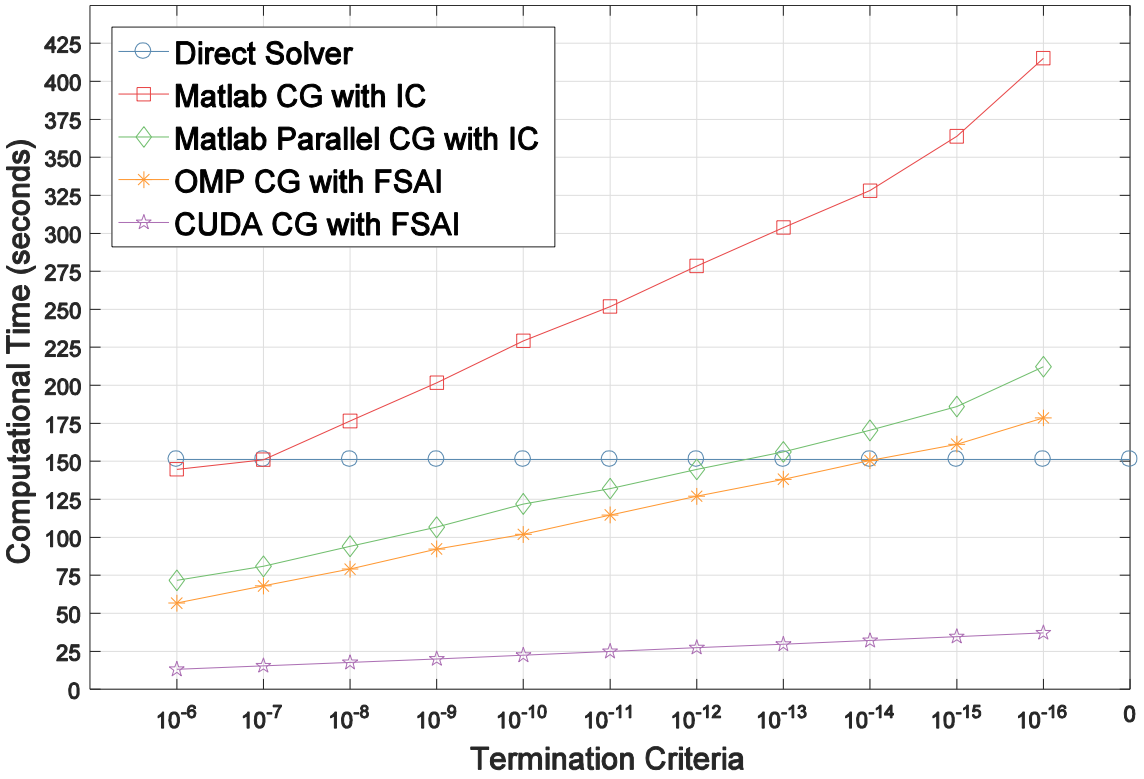

Fig. 5 Computational time as a function of termination criteria, comparison between different linear solvers. Simulation in continuous wave, for 158 sources in a 400,000 nodes mesh.

implementations that do not require much additional time to converge to smaller errors can

increase the accuracy of the estimated light propagation model whilst keeping the computational time low.

Each source is represented by one right-hand side vector in the linear system resulting from the FEM Equation (1), and the fluence must be calculated for all sources. To achieve this, the iterative solvers must create the Krylov space under the projections of each right-hand side vector, which, as expected increases the computational cost and therefore the computational time required. Fig. 6 demonstrates how the number of sources (right hand side vectors) affects the computational time of the solution, showing that the computational time increases linearly with the number of sources. It is interesting to note that the direct solver, that yields the exact solution relying on Cholesky decomposition followed by forward and backward substitutions,

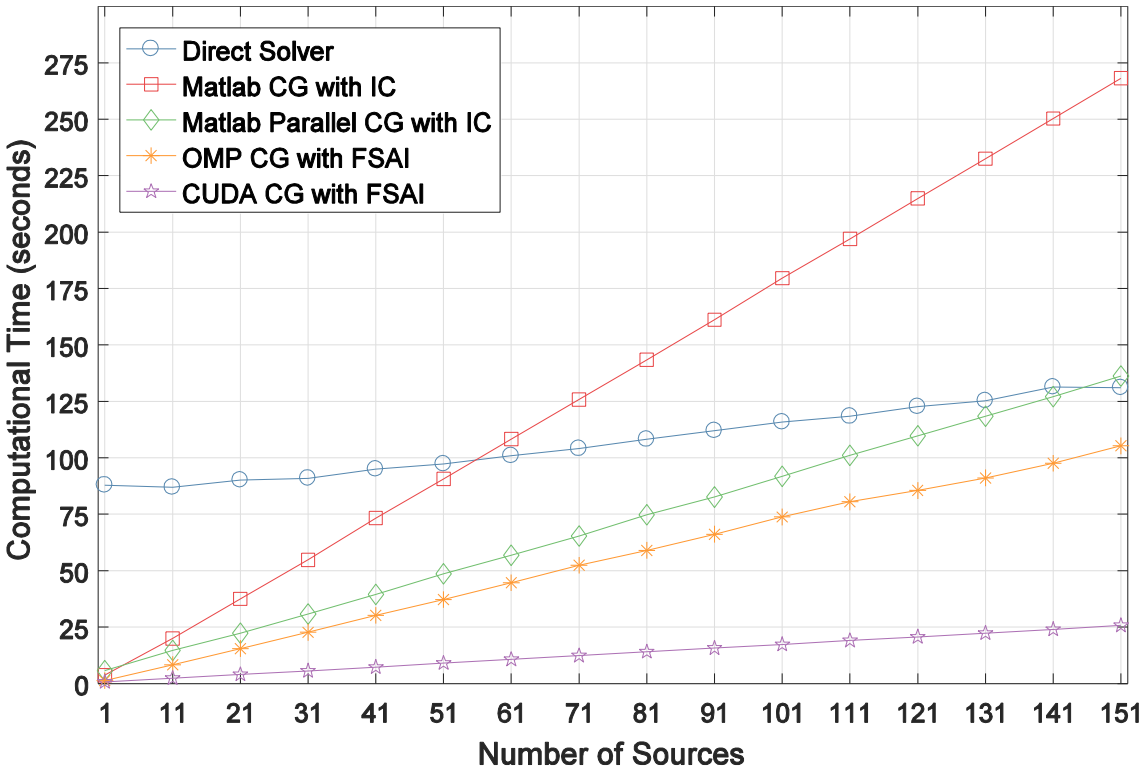

Fig. 6 Computational time as a function of excitation sources number, comparison between different linear solvers. Simulation in continuous wave, in a 400,000 nodes mesh, with 10^{-12} termination criteria.

is almost as efficient for each additional source as the GPU based solver. However, the time spent initially for the factorisation is very large, which in combination with the very high

memory requirements as discussed in section 2.3.1, render the direct solver impractical.

Nevertheless, the factor that affects computational time the most is the resolution of the mesh. The more nodes a mesh contains, the bigger the linear systems that needs to be solved is, therefore more mathematical operations have to be applied to create the Krylov space. Fig. 7 presents the computational time needed as the mesh resolution increases. The fastest of the solvers is the CUDA based solver, which achieves computational time of ~42 seconds for calculating the fluence for all 158 excitation sources in a 600,000 node mesh, this is ~0.25 second to calculate the fluence for one source. The CUDA based solver performs almost 11 times faster than the MATLAB based iterative solver without any parallelization, which takes ~460 seconds for the same calculation. The direct solver can only solve up to systems with

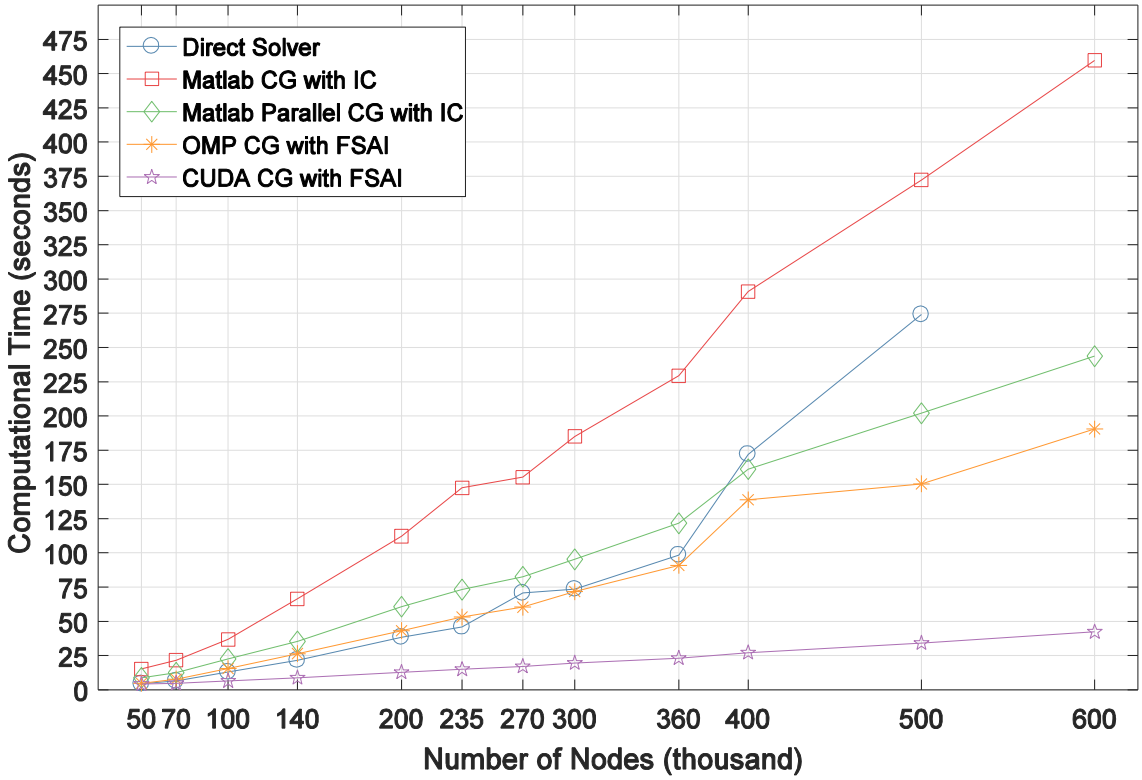

Fig. 7 Computational time with respect to mesh resolution, comparison between different linear solvers. Simulation in continuous wave, for 158 sources with 10^{-12} termination criteria.

500,000 nodes before the 16GB hardware memory availability becomes an underlying issue.

Fig. 8 displays the computational time for different mesh resolutions for frequency domain simulations at a modulation frequency of 100 MHz. The direct solver can only handle up to

200,000 nodes, due to the increased memory requirements for storing complex numbers. The displayed computational time includes the computations to create the equivalent real system and transform the fluence back to complex after solving the system where is necessary.

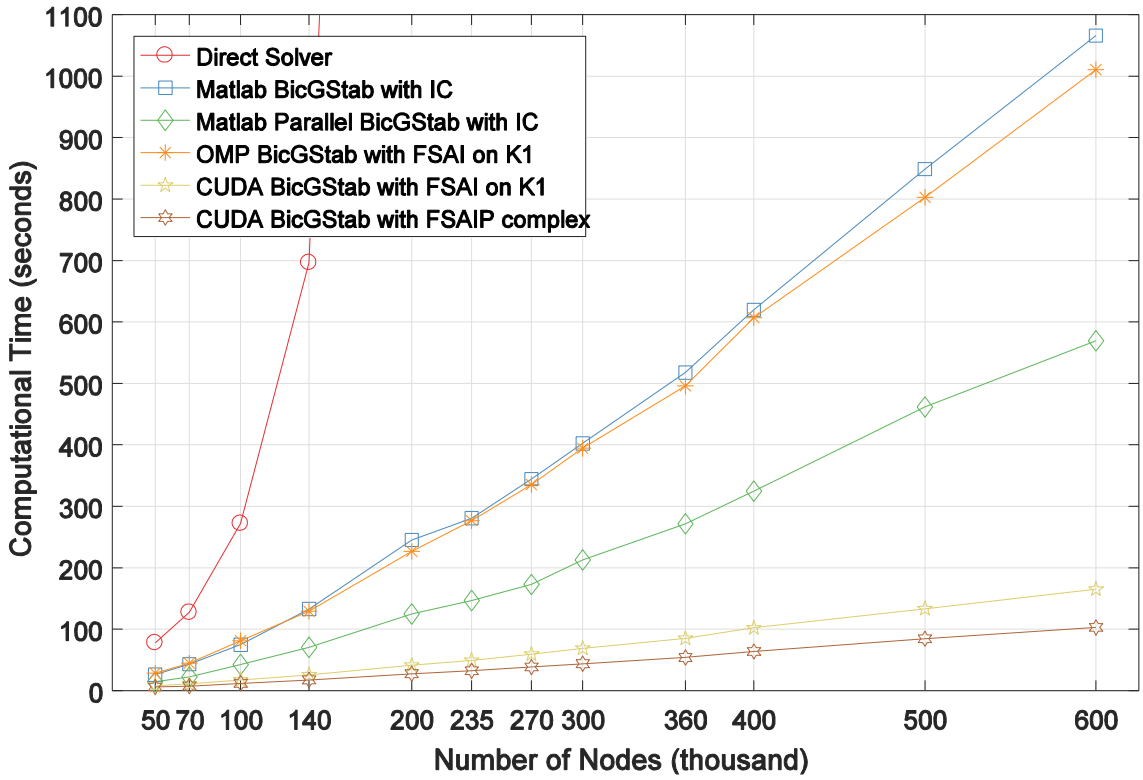

Fig. 8 Computational time with respect to mesh resolution, comparison between different linear solvers. Simulation in 100 MHz frequency, for 158 sources with 10^{-12} termination criteria.

The direct solver takes 4,612 seconds to calculate the fluence for the 200,000 nodes mesh, however Fig.8 was limited to 1,100 seconds to provide a better scale. The direct solver becomes intractable due to the increased memory requirements for complex arithmetic storage and because of the non-Hermitian nature of the FEM matrix, which is also reflected as increased memory and computational requirements for the required LU decomposition (in comparison with the Cholesky for the real cases).A linear system resulting from a frequency domain FEM mesh does not have the same condition number as the same mesh in continuous wave, due to different attenuation coefficients for frequency modulated light, which makes the FD problem harder to solve, therefore, there is not a direct analogy between their computational costs.

Though, one can roughly assume that for a given mesh if a CW solution requires O operations the FD will require $2O$. This is confirmed on our demonstrated results in Figs 6 and 8.

Furthermore, the "OMP BiCGStab with FSAI on K1" operates on the complex to real transformed (K1) matrix, resulting in double computations in comparison to the "Matlab BiCGStab with IC" which operates directly on the complex domain. As an approximation one could assume that if a mesh in CW requires O number of operations, it requires $2O$ in the complex domain but $4O$ when the complex to real transformation is used. Also, the Matlab parallel version requires almost half the computational time of the non-parallel Matlab version, and the OpenMP version is slightly faster than the Matlab parallel version when operating in the same space (O). Then it is possible to observe the following: a solution on CW would take T seconds for Matlab non-parallel, $T/2$ for Matlab parallel, and slightly faster than $T/2$ for OpenMP (note that all these cases do O operations). In contrast a solution on FD domain would take $2T$ for Matlab non-parallel (operates in $2O$), T for Matlab parallel (operates in $2O$) and slightly faster than $2T$ for OpenMP (operates on $4O$). Furthermore, the implemented complex CUDA version, which operates on $2O$, requires approximately half the computational time in comparison with the CUDA on the K1 ($4O$) space.

4. CONCLUSION

DOT is a promising imaging modality, steadily gaining ground amongst the established imaging techniques. The harmless and patient friendly procedure enables use in applications where other techniques are inadequate. However, the DOT reconstruction algorithm, especially when employed for functional brain imaging, suffers from large computational time, mainly due to solving large sparse linear systems. This work provides fast GPU and CPU implementations of efficient and stable linear solvers, based on CUDA and OpenMP respectively, compiled as mex files, to be directly accessible from MATLAB and will become publicly available in the next release of the NIRFAST package (www.nirfast.org). It is shown that numerical errors introduced

by iterative solvers are spatially located away from the excitation source. However, the distance of the numerical errors from the excitation source is related to the termination criteria, indicating that choosing large termination criteria to accelerate the modelling procedure, could negatively affect the quality of the reconstruction, depended on the application. Nevertheless, if the application allows, the computational time of any iterative solver can be greatly reduced by increasing the termination criteria. For example, for the models examined in this work, increasing the termination to 10^{-6} from 10^{-12} will reduce the computational time by half, but will increase the modelling error above 1% for the farthest SD separation. Therefore, the underlying physics, and the modelling and reconstruction procedure must be considered before attempting to solve with higher termination criteria. However, it is now computationally feasible to select lower termination criteria for the iterative solvers, practically eliminating any error induced by the approximate solving or the complexity of the volume, as the GPU parallelized approach has overly significantly lower computational time. Furthermore, the proposed approaches can be very efficient for systems with large number of sources and detectors since the computational time is not greatly affected by solving for multiple sources; and in addition, can be employed in frequency domain simulations. Based on the performed experiments, the fastest approach is to parallelize the matrix to vector operations involved in iterative solvers in GPU architectures.

The produced solvers allow researchers to explore new approaches in DOT, that until now were out of reach due to the slowness of the algorithm. Simulations of light propagation that would take a long time, now can be done in a few minutes, forging a path towards real-time DOT. The work presented here is based on systems with one GPU node; though, the same philosophy can be applied in systems with multiple GPUs and extended to cloud computing to achieve real-time solutions. Parallelization approaches can also be applied for the optimization of the inverse problem of DOT, where the creation and the inversion of the Jacobian are currently the most computationally expensive parts of the algorithm, especially when recovering

absolute optical parameters. In functional brain imaging, creating sparse Jacobians enables to solve the linear inverse problem directly in the GPU in real-time speed for each temporal set of measurements.

5. DISCLOSURES

The authors have no relevant financial interests in this article and no potential conflicts of interest to disclose.

6. ACKNOWLEDGMENTS

This work has been funded by the National Institutes of Health (NIH) Grant R01EB009233-2, RO1-CA132750 and Autism Speaks Meixner Translational Postdoctoral Fellowship 7962.

REFERENCES

1. Craig AD. How do you feel? Interoception: the sense of the physiological condition of the body. *Nat Rev | Neurosci*. 2002;3(August).
2. Sitaram R, Veit R, Stevens B, et al. Acquired Control of Ventral Premotor Cortex Activity by Feedback Training: An Exploratory Real-Time fMRI and TMS Study. *Neurorehabil Neural Repair*. 2012. doi:10.1177/1545968311418345.
3. Ruiz S, Lee S, Soekadar SR, et al. Acquired Self-control of Insula Cortex Modulates Emotion Recognition and Brain Network Connectivity in Schizophrenia. *Wiley Period Inc Hum Brain Mapp*. 2011;0(July). doi:10.1002/hbm.21427.
4. Zeff BW, White BR, Dehghani H, Schlaggar BL, Culver JP. Retinotopic mapping of adult human visual cortex with high-density diffuse optical tomography. *Proc Natl Acad Sci U S A*. 2007;104(29):12169-12174. doi:10.1073/pnas.0611266104.
5. Boas D a., Chen K, Grebert D, Franceschini MA. Improving the diffuse optical imaging spatial resolution of the cerebral hemodynamic response to brain activation in humans. *Opt Lett*. 2004;29(13):1506-1508. doi:10.1364/OL.29.001506.
6. Austin T, Gibson AP, Branco G, et al. Three dimensional optical imaging of blood volume and oxygenation in the neonatal brain. *Neuroimage*. 2006;31(4):1426-1433. doi:10.1016/j.neuroimage.2006.02.038.
7. Liao SM, Ferradal SL, White BR, Gregg N, Inder TE, Culver JP. High-density diffuse optical tomography of term infant visual cortex in the nursery. *J Biomed Opt*. 2012;17(8):81414. doi:10.1117/1.JBO.17.8.081414.
8. Leung TS, Tachtsidis I, Smith M, Delpy DT, Elwell CE. Measurement of the absolute optical properties and cerebral blood volume of the adult human head with hybrid differential and spatially resolved spectroscopy. *Phys Med Biol*. 2006;51(3):703-717. doi:10.1088/0031-9155/51/3/015.
9. Dehghani H, Eames ME, Yalavarthy PK, et al. Near infrared optical tomography using NIRFAST: Algorithm for numerical model and image reconstruction. *Commun Numer Methods Eng*. 2008;25(6):711-732. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2826796&tool=pmcentrez&rendertype=abstract. Accessed March 24, 2016.
10. Scherl H, Keck B, Kowarschik M, Hornegger J. Fast GPU-based CT reconstruction using the Common Unified Device Architecture (CUDA). *IEEE Nucl Sci Symp Conf Rec*. 2007;6:4464-4466. doi:10.1109/NSSMIC.2007.4437102.

11. Yoo S-H, Park J-H, Jeong C-S. Accelerating Multi-scale Image Fusion Algorithms Using CUDA. *Soft Comput Pattern Recognition, 2009 SOCPAR '09 Int Conf*. 2009:278-282. doi:10.1109/SoCPaR.2009.63.
12. Zhang Z, Miao Q, Wang Y. CUDA-based Jacobi's iterative method. *IFCSTA 2009 Proc - 2009 Int Forum Comput Sci Appl*. 2009;1:259-262. doi:10.1109/IFCSTA.2009.68.
13. Ren N, Liang J, Qu X, Li J, Lu B, Tian J. GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues. *Opt Express*. 2010;18(7):6811-6823. doi:10.1364/OE.18.006811.
14. Fang Q, Boas DA. Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units. 2009;17(22):20178-20190.
15. Alerstam E, Svensson T, Andersson-Engels S. Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration. *J Biomed Opt*. 2008;13(6):60504. doi:10.1117/1.3041496.
16. Prakash J, Chandrasekharan V, Upendra V, Yalavarthy PK. Accelerating frequency-domain diffuse optical tomographic image reconstruction using graphics processing units. *J Biomed Opt*. 2010;15(6):66009. doi:10.1117/1.3506216.
17. Zhang T, Zhou J, Carney PR, Jiang H. Towards real-time detection of seizures in awake rats with GPU-accelerated diffuse optical tomography. *J Neurosci Methods*. 2015;240:28-36. doi:10.1016/j.jneumeth.2014.10.018.
18. Saikia MJ, Kanhirodan R. High Performance Single and Multi-GPU Acceleration for Diffuse Optical Tomography. In: *International Conference on Contemporary Computing and Informatics, IC3I 2014*. ; 2014:1320-1323.
19. Yi X, Wang X, Chen W, Wan W, Zhao H, Gao F. Full domain-decomposition scheme for diffuse optical tomography of large-sized tissues with a combined CPU and GPU parallelization. *Appl Opt*. 2014;53(13):2754-2765. doi:10.1364/AO.53.002754.
20. Schweiger M. GPU-accelerated finite element method for modelling light transport in diffuse optical tomography. *Int J Biomed Imaging*. 2011;2011. doi:10.1155/2011/403892.
21. Carpenter CM, Pogue BW, Jiang S, et al. Image-guided optical spectroscopy provides molecular-specific information in vivo: MRI-guided spectroscopy of breast cancer hemoglobin, water, and scatterer size. *Opt Lett*. 2007;32(8):933-935. doi:10.1364/OL.32.000933.
22. Brooksby B, Jiang S, Dehghani H, et al. Magnetic resonance-guided near-infrared tomography of the breast. *Rev Sci Instrum*. 2004;75(12):5262-5270. doi:10.1063/1.1819634.
23. Mazziotta J, Toga A, Evans A, et al. A probabilistic atlas and reference system for the human brain: International Consortium for Brain Mapping (ICBM). *Philos Trans R Soc Lond B Biol Sci*. 2001;356(1412):1293-1322. doi:10.1098/rstb.2001.0915.
24. Jermyn M, Ghadyani H, Mastanduno MA, et al. Fast segmentation and high-quality three-dimensional volume mesh creation from medical images for diffuse optical tomography. *J Biomed Opt*. 2013;18(8):86007. doi:10.1117/1.JBO.18.8.086007.
25. Paulsen KD, Jiang H. Spatially varying optical property reconstruction using a finite element diffusion equation approximation. *Med Phys*. 1995;22(6):691-701. doi:10.1118/1.597488.
26. Arridge SR, Schweiger M, Hiraoka M, Delpy DT. A finite element approach for modeling photon transport in tissue. *Med Phys*. 1993;20(2 Pt 1):299-309. doi:10.1118/1.597069.
27. Hueber DM, Franceschini M a, Ma HY, et al. Non-invasive and quantitative near-infrared haemoglobin spectrometry in the piglet brain during hypoxic stress, using a frequency-domain multidistance instrument. *Phys Med Biol*. 2001;46(1):41-62. doi:10.1088/0031-9155/46/1/304.
28. Arridge SR, Lionheart WRB. Nonuniqueness in diffusion-based optical tomography. *Opt Lett*. 1998;23(11):882-884.
29. Zhan Y, Eggebrecht AT, Culver JP, Dehghani H. Image quality analysis of high-density diffuse optical tomography incorporating a subject-specific head model. *Front Neuroenergetics*. 2012;(MAY). doi:10.3389/fnene.2012.00006.
30. White BR, Culver JP. Quantitative evaluation of high-density diffuse optical tomography: in vivo resolution and mapping performance. *J Biomed Opt*. 2010;15(2):26006. doi:10.1117/1.3368999.
31. Wu X, Eggebrecht AT, Ferradal SL, Culver JP, Dehghani H. Quantitative evaluation of atlas-

based high-density diffuse optical tomography for imaging of the human visual cortex. *Biomed Opt Express*. 2014;5(11):3882-3900. doi:10.1364/BOE.5.003882.

32. Eggebrecht AT, White BR, Ferradal SL, et al. A quantitative spatial comparison of high-density diffuse optical tomography and fMRI cortical mapping. *Neuroimage*. 2012;61(4):1120-1128. doi:10.1016/j.neuroimage.2012.01.124.
33. Eggebrecht AT, Ferradal SL, Robichaux-Viehoever A, et al. Mapping distributed brain function and networks with diffuse optical tomography. *Nat Photonics*. 2014;8(6):448-454. doi:10.1038/nphoton.2014.107.
34. Arridge SR, Schweiger M. Photon-measurement density functions. Part 2: Finite-element-method calculations. *Appl Opt*. 1995;34(34):8026-8037. doi:10.1364/AO.34.008026.
35. Mathworks. spparams. https://uk.mathworks.com/help/matlab/ref/spparms.html. Published 2017. Accessed October 5, 2017.
36. Saad Y. Parallel iterative methods for sparse linear systems. *Stud Comput Math*. 2001;8(C):423-440. doi:10.1016/S1570-579X(01)80025-2.
37. Nachtigal M, Reddy SC, Trefethen LN. HOW FAST ARE NONSYMMETRIC MATRIX ITERATIONS?* NOeL. *SIAM J MATRIX ANAL APPL*. 1992;13(3):778-795.
38. Faberf V, Manteuffel T. Necessary and sufficient conditions for the existance of a conjugate gradient method. *SIAM J Numer Anal*. 1984;21(2):352–362. doi:10.1080/03081087.2013.851198.
39. Axelsson O, Farouq S, Neytcheva M. *Comparison of Preconditioned Krylov Subspace Iteration Methods for PDE-Constrained Optimization*. Numerical Algorithms; 2016. doi:10.1007/s11075-016-0111-1.
40. Benzi M. Preconditioning Techniques for Large Linear Systems: A Survey. *J Comput Phys*. 2002;182(2):418-477. doi:http://dx.doi.org/10.1006/jcph.2002.7176.
41. Elman H, Howle VE, Shadid J, Shuttleworth R, Tuminaro R. Block preconditioners based on approximate commutators. *SIAM J Sci Comput*. 2006;27(5):1651-1668. doi:10.1137/040608817.
42. Saad Y, Zhang J. Enhanced multi-level block ILU preconditioning strategies for general sparse linear systems. *J Comput Appl Math*. 2001;130(1):99-118. doi:10.1016/S0377-0427(99)00388-X.
43. Kolotilinat LY, Yeremin AY. Factorized Sparse Approximate Inverse Preconditionings I. Theory*. *Siam J Matrix Anal Appl*. 1993;14(1):45-58. doi:10.1137/0614004.
44. Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming. *IEEE Comput Sci Eng*. 1998;5(1):46-55. doi:10.1109/99.660313.
45. Nickolls J, Buck I, Garland M, Skadron K. Scalable parallel programming with CUDA. *AMC Queue*. 2008;6(April):40-53. doi:10.1145/1365490.1365500.
46. Labs P. PARALUTION - v1.0.0. https://www.paralution.com/. Published 2015.
47. Tillet P, Rupp K. Towards Performance-Portable, Scalable, and Convenient Linear Algebra. ... *HotPar'13*. 2013. http://0b4af6cdc2f0c5998459-c0245c5c937c5dedcca3f1764ecc9b2f.r43.cf2.rackcdn.com/11423-hotpar13-tillet.pdf.
48. Day D, Heroux MA. SOLVING COMPLEX-VALUED LINEAR SYSTEMS VIA EQUIVALENT REAL FORMULATION. 2001;23(2):480-498.
49. Humphrey JR, Price DK, Spagnoli KE, Paolini AL, Kelmelis EJ. CULA: hybrid GPU accelerated linear algebra routines. *Defense*. 2010;7705(1):770502-770502-770507. doi:10.1117/12.850538.